Building Automator Actions for System Administrators

Session 645

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

Building Automator Actions for System Administrators

Steve Hayman
Consulting Engineer
shayman@apple.com

http://developer.apple.com/wwdc2005/

Introduction

Learn about using Automator with sh, perl and other scripting languages

What You'll Learn

- Making use of Run Shell Script
- Building new actions with AMShellScriptAction
- Mixing and matching shell scripts with other actions

What You Won’t Learn

In this session at least

- Details of sh, perl, ruby or other languages
- Writing Applescript or Cocoa stages
- Every little detail of Interface Builder or XCode

Quick Review of Automator

And a demo

http://developer.apple.com/wwdc2005/

What’s New and Improved

Improved already, can you believe it

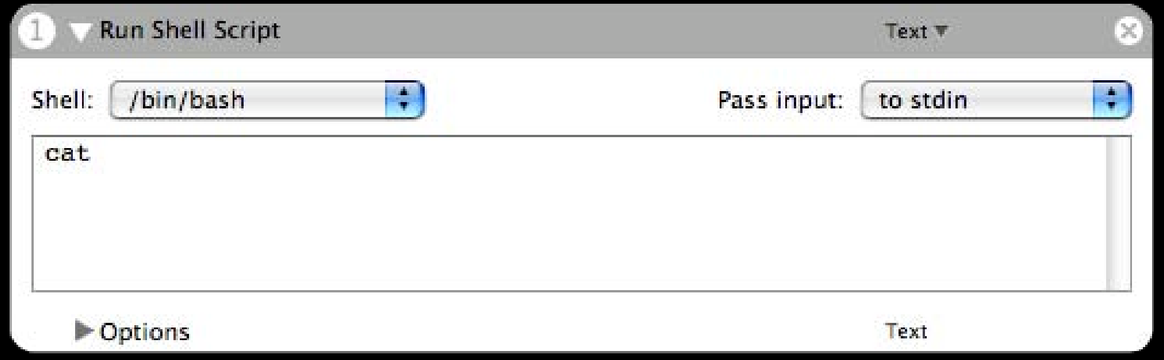
http://developer.apple.com/wwdc2005/

This Just In

- Improved **Run Shell Script** action
- New **AMShellScriptAction** project template

Run Shell Script

Shipping in Tiger

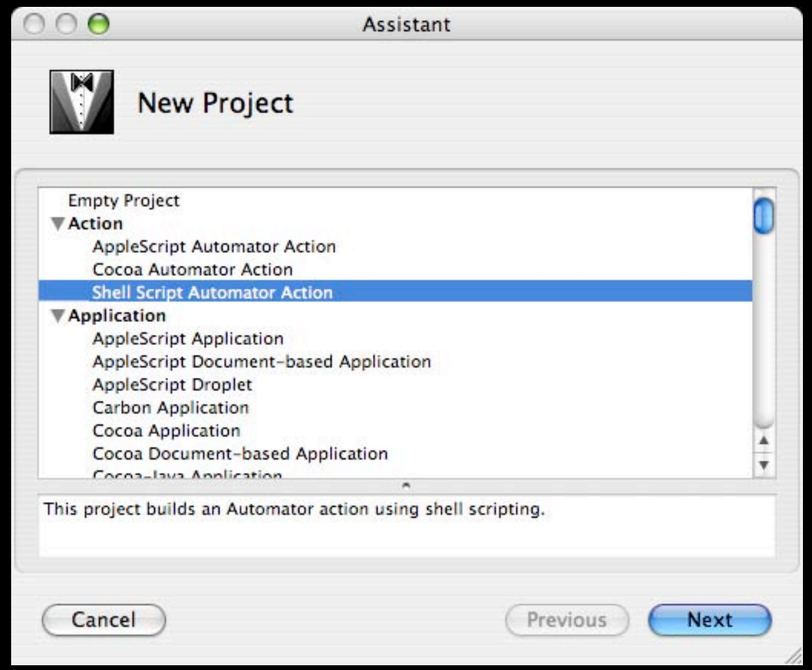


Available in Xcode 2.1

- Improved Run Shell Script.action available with Xcode 2.1

AMShellScriptAction

- XCode 2.1 project template
- Create brand new actions with GUIs using shell scripts, Perl, python, etc.



Anatomy of an Action

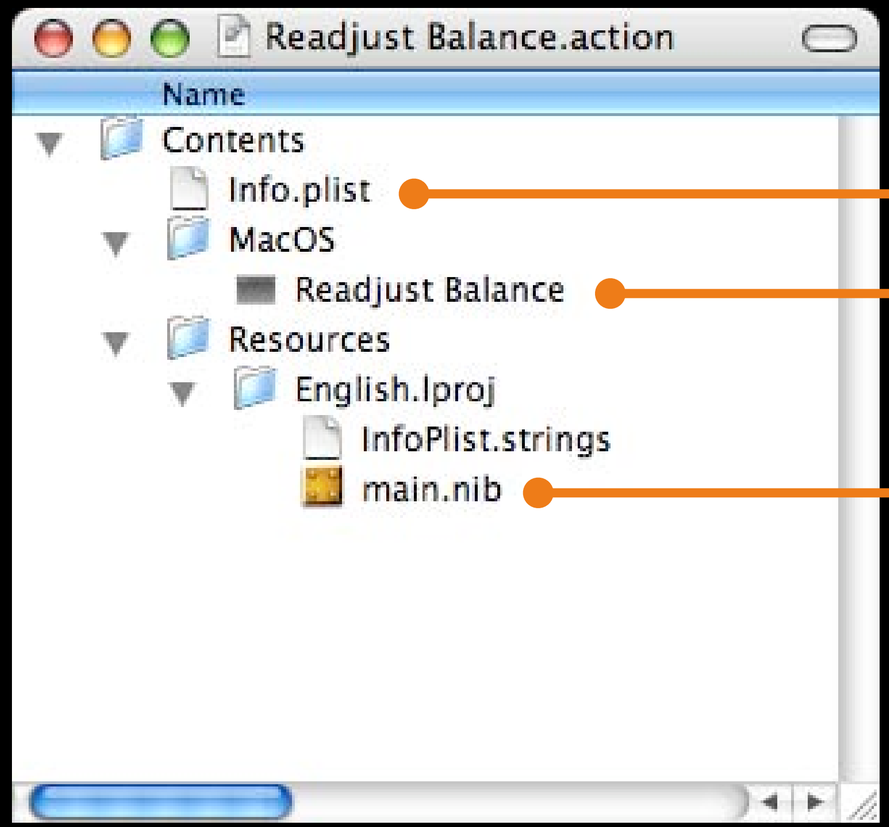
http://developer.apple.com/wwdc2005/

Goal of Automator Workflows

Choose Actions and string them together

- Over 150 actions supplied with Automator
- Very flexible Run Shell Script and Run Applescript actions
  - Just type in a script
- Or, create your own!
  - Create a script and a GUI for it

Contents of a Cocoa Action

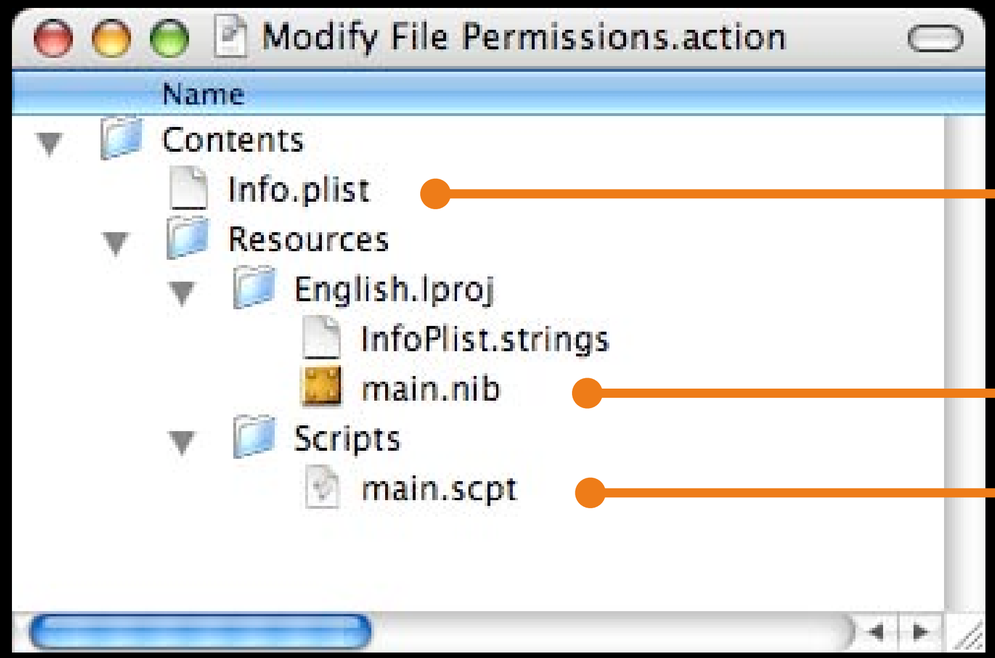


Description of the action

Compiled Code

User Interface

Contents of an Applescript Action

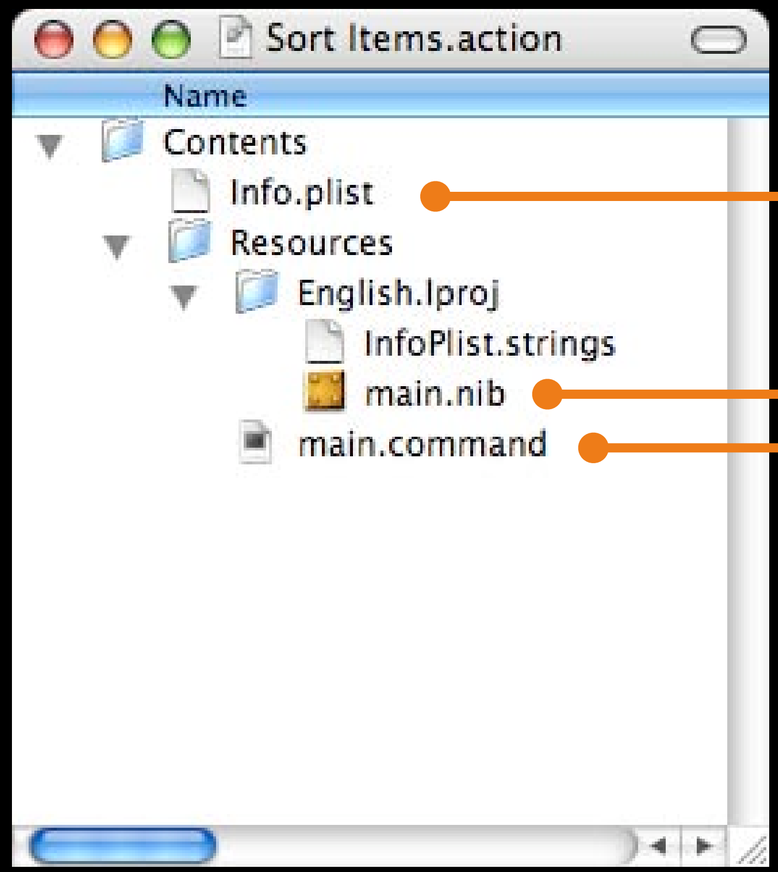


Description of the action

User Interface

User AppleScript

Contents of a Shell Action



Description of the action

User Interface

Shell Commands

What's in an Action?

- What application it belongs to
  - Terminal, Mail, TextEdit, Automator…
- What sort of input it takes
  - Text, Files, Images, Mail Messages, Address Book Items…
- What sort of output it provides
  - Text, Files, Images, Mail Messages, Address Book Items…

Info.plist

- Something to execute
  - Compiled Objective-C code
  - Applescripts
  - Shell scripts

What Kind of Actions Do We Script Lovers Like?

- Actions associated with Terminal
  - (just a convention)
- Actions that get text as input
- Actions that provide text as output
- Scripts that process the text

Does This Remind You of UNIX Pipes?

- It should!
- Very first UNIX pipe I ever saw in 1981

```
cat file |
tr -cs ‘A-Za-z’ ‘\012’ |
tr A-Z a-z |
sort |
uniq |
comm -23 - /usr/share/dict/words
```

- You could do that in Automator today

Demo

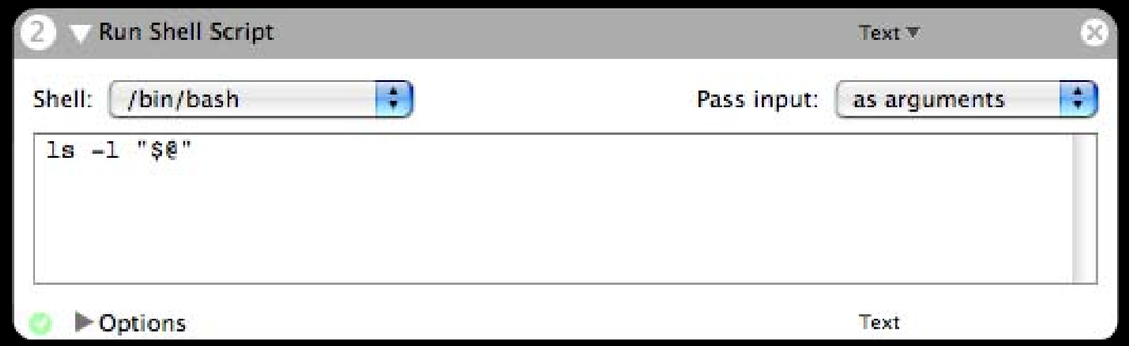
Of doing exactly that in Automator today

http://developer.apple.com/wwdc2005/

Inner Workings of Shell Actions

http://developer.apple.com/wwdc2005/

Run Shell Script



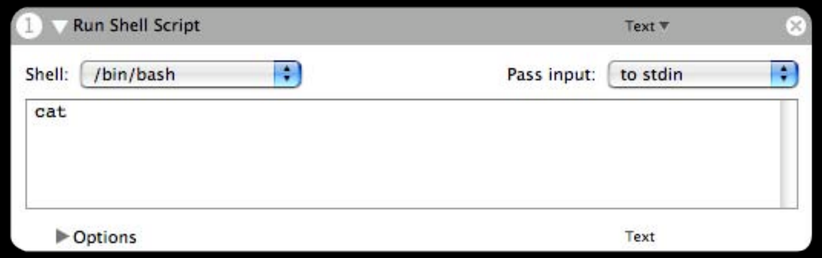
- Text from previous command is collected
- Script is executed
  - Previous text passed as stdin, or as arguments
- Output from script is collected and passed to next action

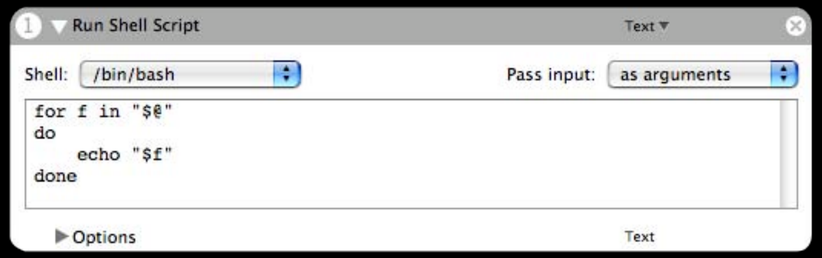
What Language Can I Use?

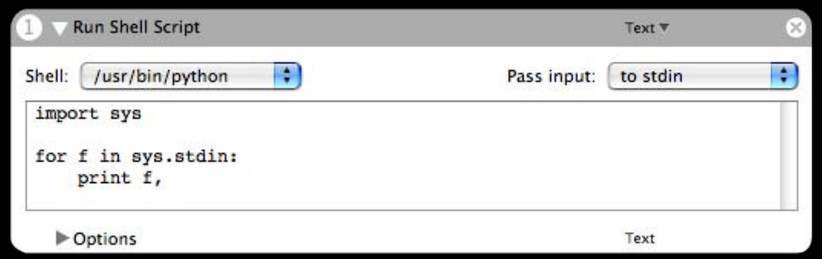
- Action knows how to launch scripts in these languages
  - bash
  - csh
  - ksh
  - sh
  - tcsh
  - zsh
  - perl
  - python
  - ruby
- Run Shell Script.action/Contents/Resources/Shells.plist

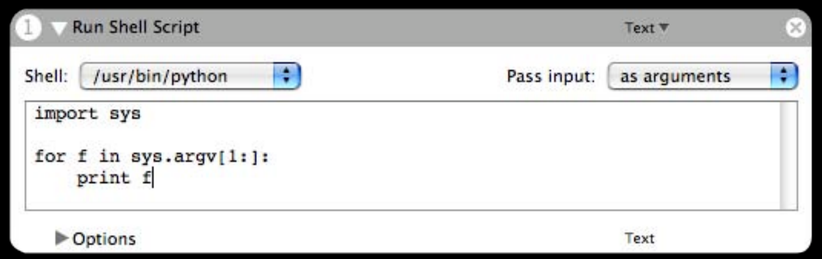
Automator Helps You Out

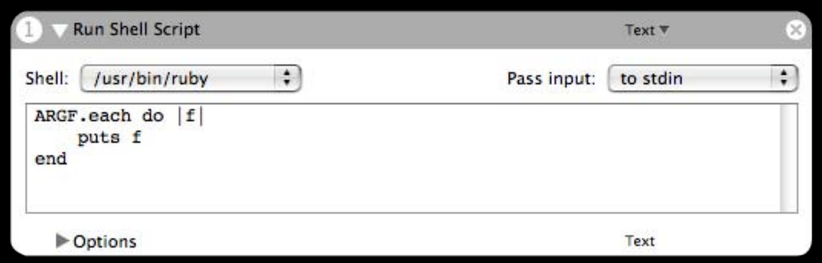
18 Script templates for 9 languages in 2 styles











Arguments vs. Standard Input

- Input to your script is a list of strings
- How do you want to handle the list?
- stdin
  - Strings are all written to the standard input of your script, separated by newlines
- arguments
  - Strings are passed as individual command-line arguments to your script

So Which Should I Use?

- Standard Input
  - Processing all the input at once
  - traditional UNIX pipes and filters

```
# Sort input lines backwards on field 4
sort -r -k 4
```

- Arguments
  - Going over each input item one at a time

```
# List attributes of each file
for aFile in "$@"; do
  ls -l "$aFile"
done
```

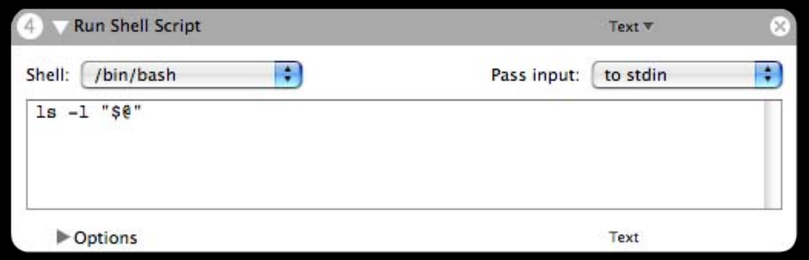
Either way they both write to standard output

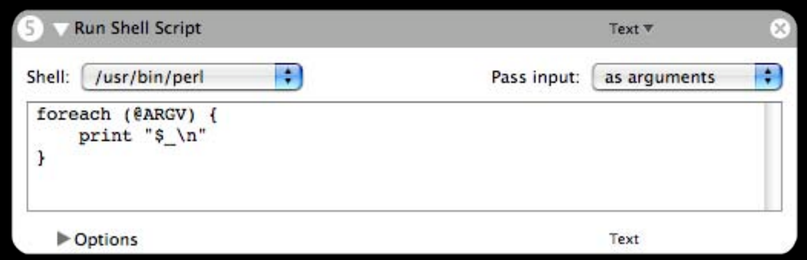
How AMShellScriptActions Work

http://developer.apple.com/wwdc2005/

Run Shell Script

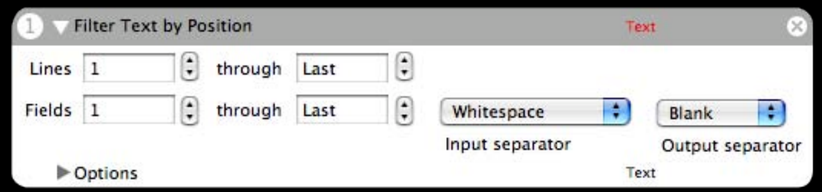
A script
Input via arguments or stdin





AMShellScriptActions

A script plus a custom GUI
Input via stdin

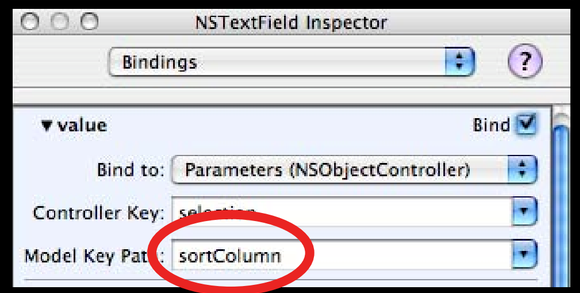


AMShellScriptActions

- Bindings in the UI are checked

main.command

```
#!/bin/sh
...
```

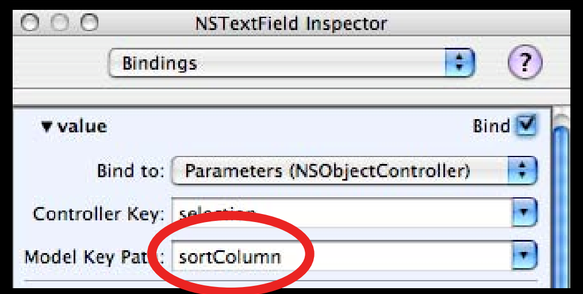


AMShellScriptActions

- A list of bindings and values is built

main.command

```
#!/bin/sh
...
```



sortColumn = 1
sortDirection = 0
sortStyle = 0
removeDuplicates = 0

AMShellScriptActions

- main.command is launched with the bindings in its Environment variables

main.command

```
#!/bin/sh
...
```

```
sortColumn = 1
sortDirection = 0
sortStyle = 0
removeDuplicates = 0
```

AMShellScriptActions

- Text from the previous action is passed to main.command as its standard input

main.command

```
#!/bin/sh
...
```

sortColumn = 1
sortDirection = 0
sortStyle = 0
removeDuplicates = 0

AMShellScriptActions

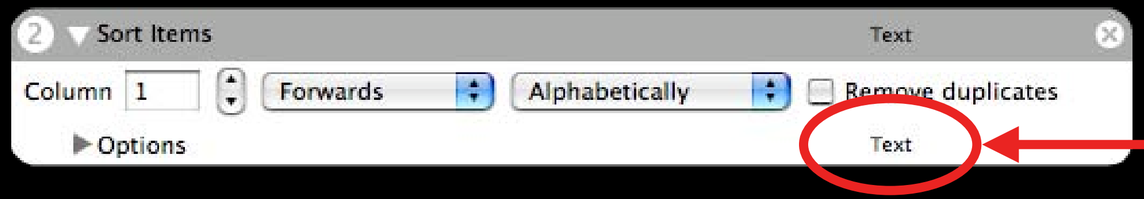
- main.command runs
  - Checks its environment vars
  - Reads from standard input
  - Writes to standard output

main.command

```
#!/bin/sh

sort -k $sortColumn…
```

sortColumn = 1
sortDirection = 0
sortStyle = 0
removeDuplicates = 0

AMShellScriptActions



- Standard output is collected and passed to the next action

main.command

```
#!/bin/sh
…
sort -k $sortColumn…
```

sortColumn = 1
sortDirection = 0
sortStyle = 0
removeDuplicates = 0

Conversions

http://developer.apple.com/wwdc2005/

Our Actions Are Text-based

- Receive text
- Do something
- Emit text
- But most other automator actions deal with other things
  - Files
  - Photos
  - Tunes

Conversion Actions

- Shell-based actions read and write text
  - “com.apple.cocoa.string”
- Other actions read and write all sorts of other things
  - Aliases, images, …
- Automator will try to insert the right conversion action magically

A Common Conversion—Files to Text

- Lots of actions produce a list of files
  - Represented internally as special Files/Folders objects
- We'd rather deal with text

Conversions Inserted Automatically

Conversion from Files to Text inserted by Automator

/System/Library/Automator Conversions—*.caction

Convert Account object to Mailbox object.caction
Convert Address Book Object to Group Object.caction
Convert Address Book Object to Person Object.caction
Convert Album object to Photo object.caction
Convert Alias object to Text object.caction
Convert Calendar object to Event object.caction
Convert Calendar object to To Do object.caction
Convert Finder object to Alias object.caction
Convert Group object to Person object.caction
Convert Mail object to Account object.caction
Convert Mailbox object to Message object.caction
Convert Photo object to Alias object.caction
Convert Playlist object to Playlist object.caction
Convert Playlist object to Song object.caction
Convert Source object to Playlist object.caction
Convert Spotlight item to Alias object.caction
Convert Spotlight item to Mail message.caction
Convert Spotlight object to Address Book object.caction
Convert Spotlight object to Font file.caction
Convert Spotlight object to Image file.caction
Convert Spotlight object to Movie file.caction
Convert Spotlight object to PDF file.caction
Convert Spotlight object to iCal event.caction
Convert Spotlight object to iTunes song.caction
Convert Text object to Alias object.caction
Convert iCal object to Calendar object.caction
Convert iPhoto object to Album object.caction
Convert iPhoto object to Photo object.caction
Convert iTunes object to Alias object.caction
Convert iTunes object to Song object.caction

Lots of Conversions to and From Text

- Finder files, folders, aliases
- iTunes items
- iPhoto items
- Spotlight items

Interesting Conversions

applescript.text-object

So Fortunately

- Your script object can read and write UNIX pathnames
- They'll be converted to the right thing
- Your action can handle much more than just plain text input

Using Existing Actions

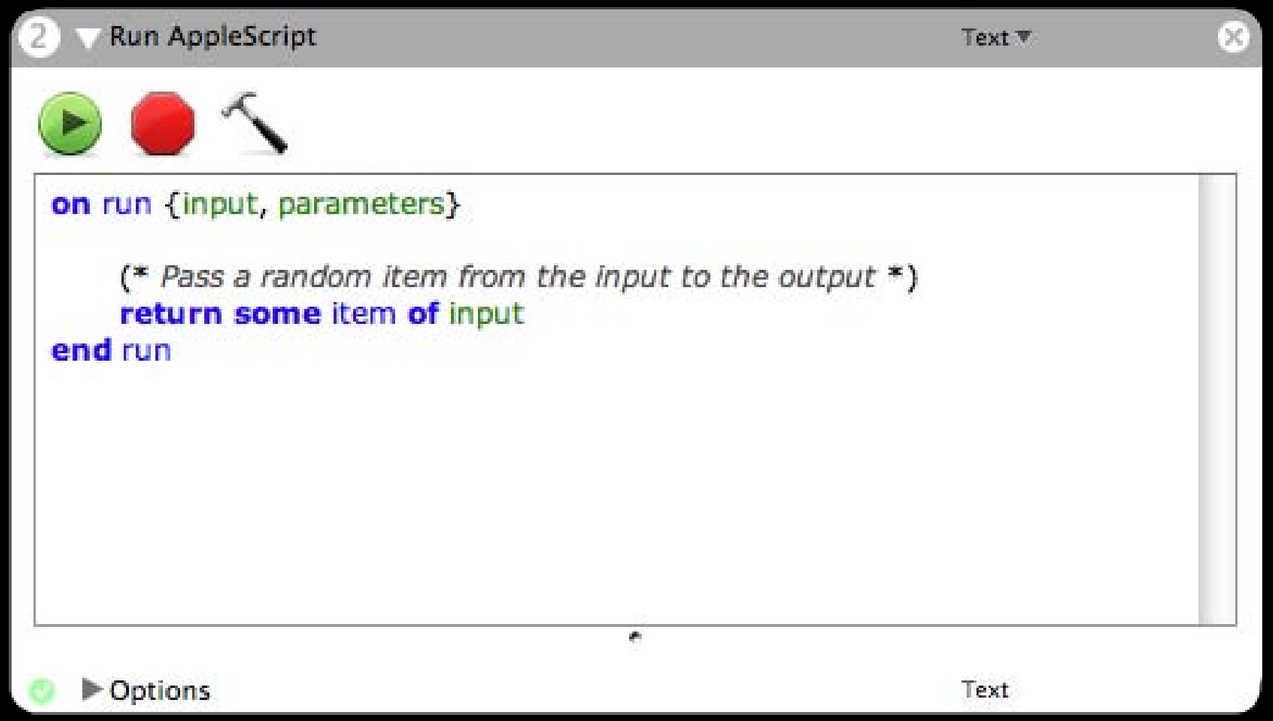
http://developer.apple.com/wwdc2005/

Run Applescript

http://developer.apple.com/wwdc2005/

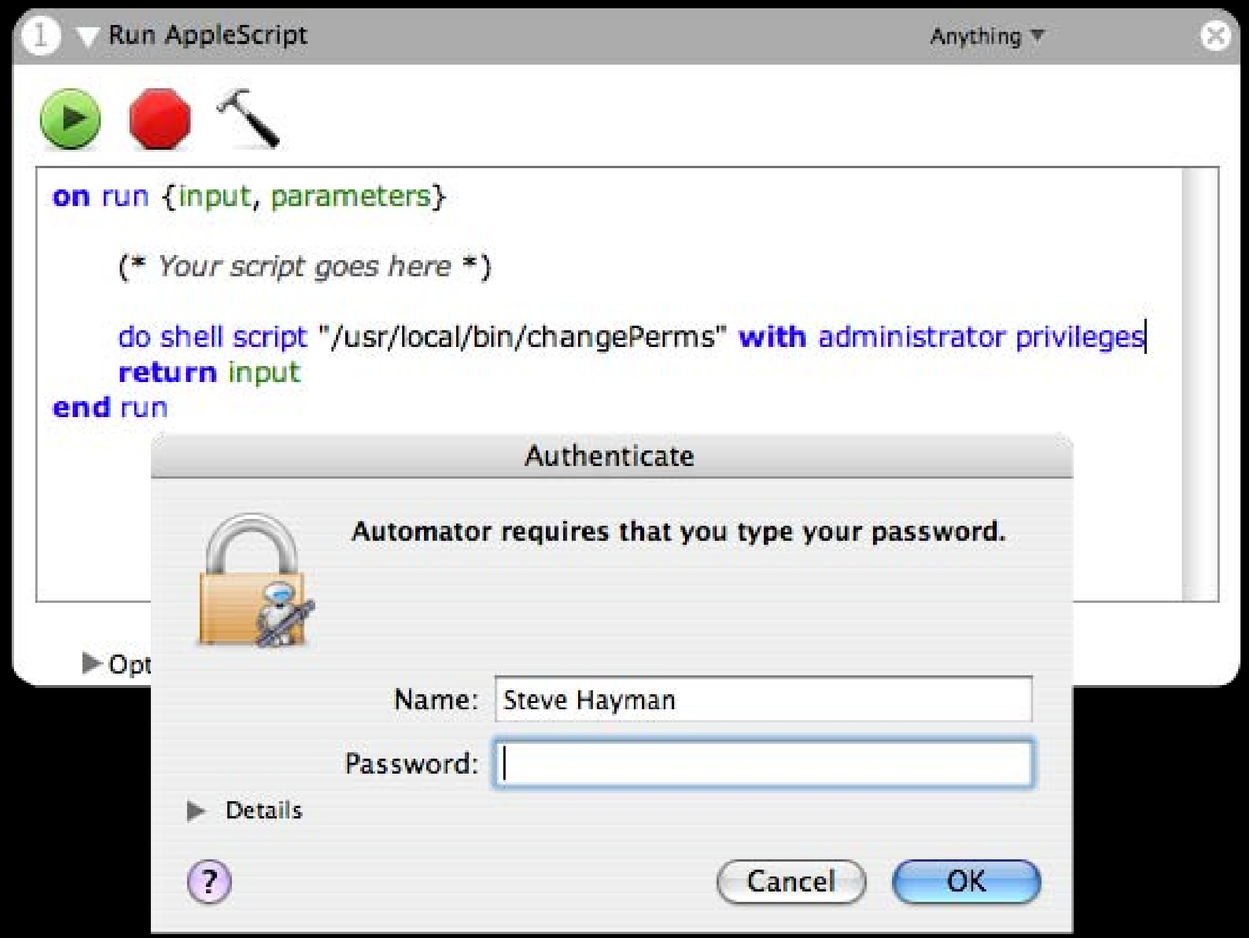
Run Applescript

You should know a little about Applescript



Useful for Authenticating a Shell Script

Do shell script “myScript” with administrator privileges

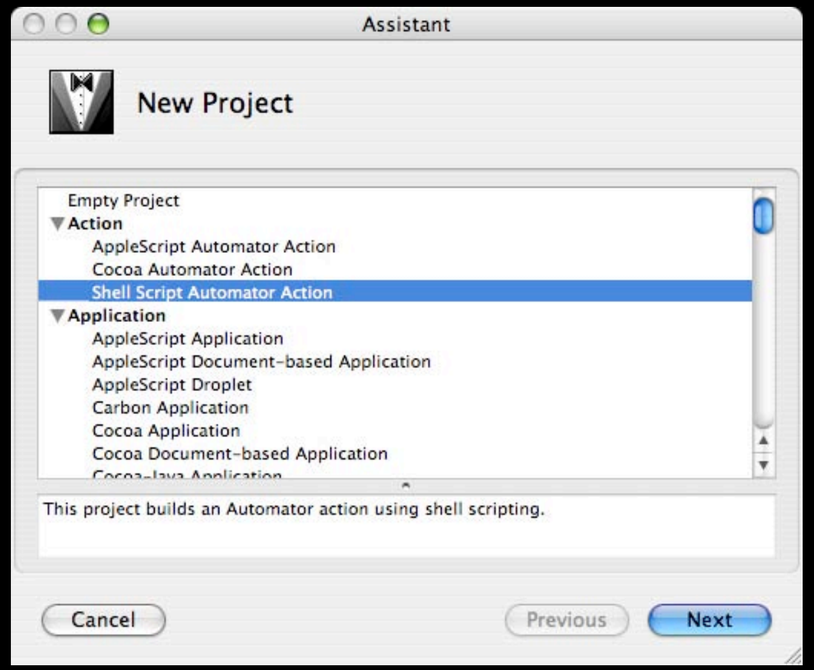


Creating Your Own Action

http://developer.apple.com/wwdc2005/

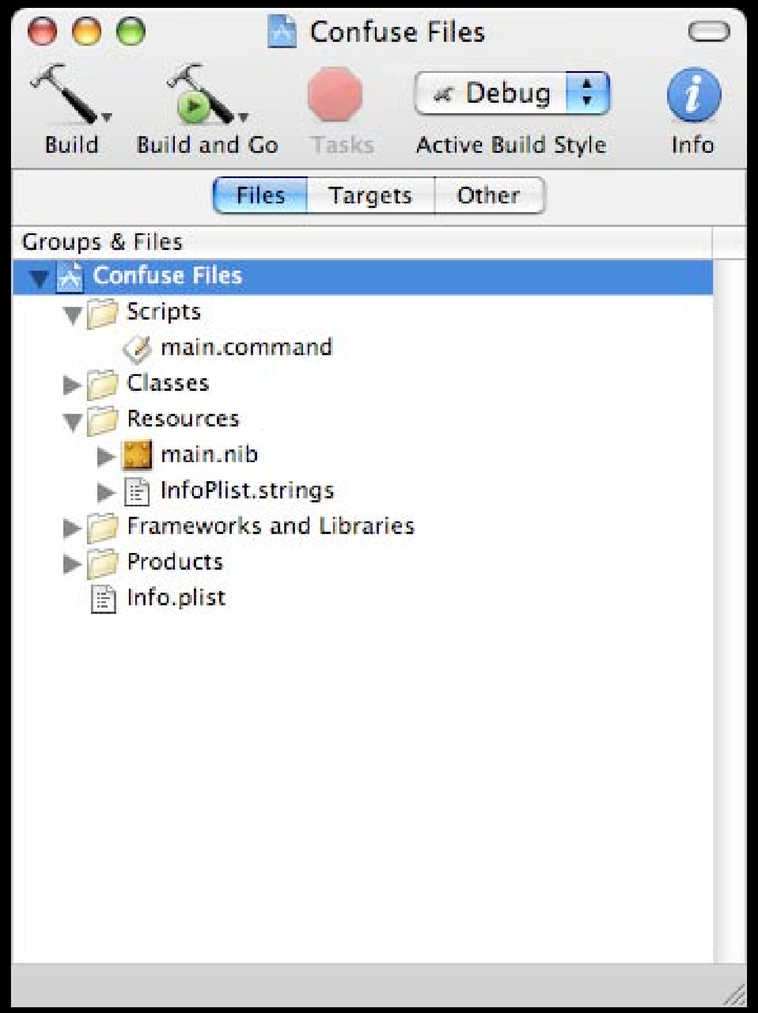
Start in XCode

- Shell Script Automator Action
- New in the latest developer tools release



Start in XCode

- main.command
  - Your custom script
- main.nib
  - Your user interface

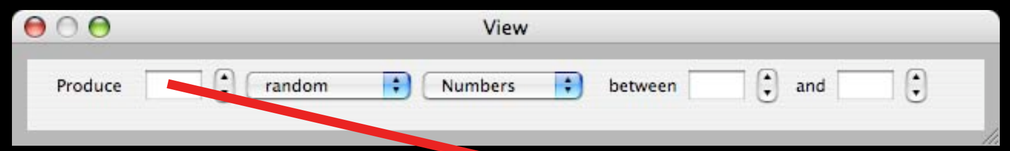


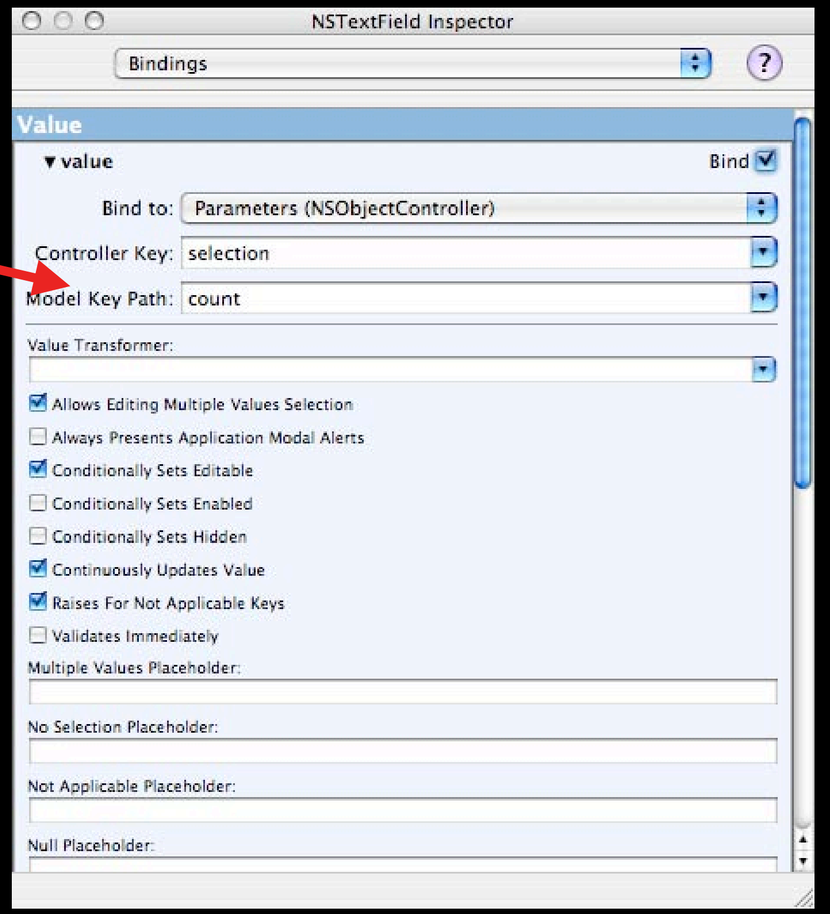
Use Interface Builder

- Edit main.nib
- Design the UI
- “Bind” the objects in the UI to the parameters of your action

AMShellScriptAction

- Design your UI in IB
- Settings will be set in your script's environment





Simple Example: Sorting

- Let’s build an action that sorts its input
  - And make it fancier with a UI

Demo:
Building a Script Action

http://developer.apple.com/wwdc2005/

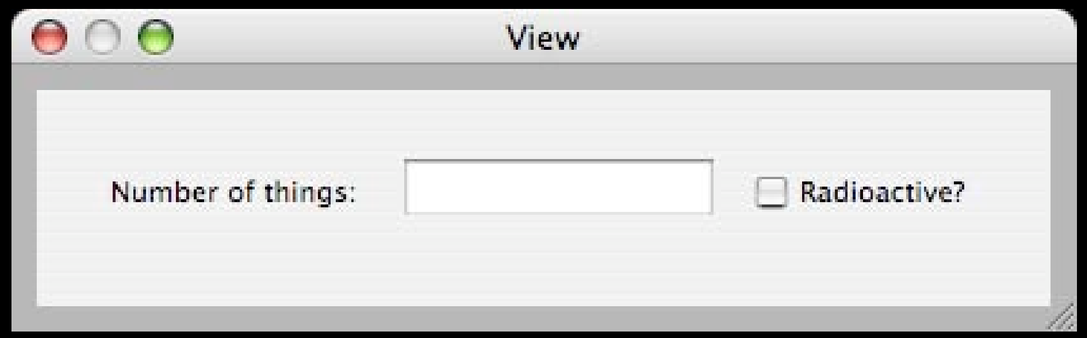
Binding Things in IB

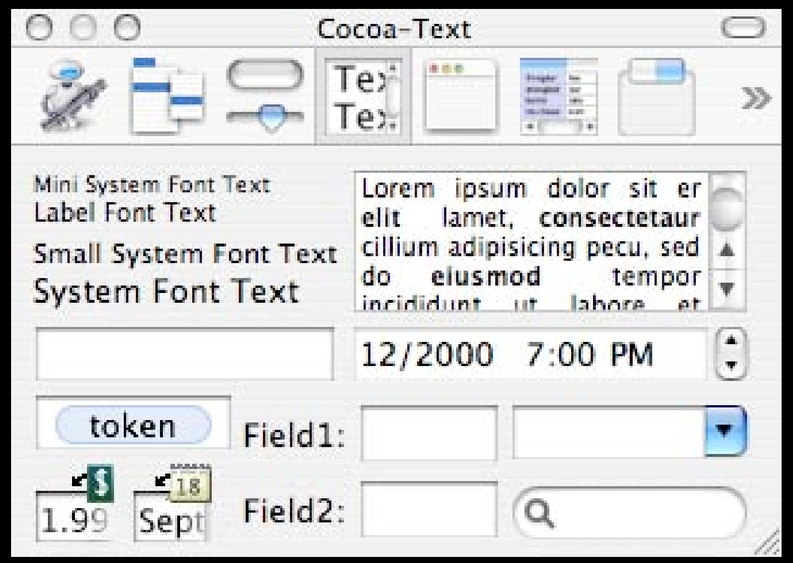
Interface Builder Is Complex

- For Automator, though, it's simple
  - (Mostly)

In Interface Builder

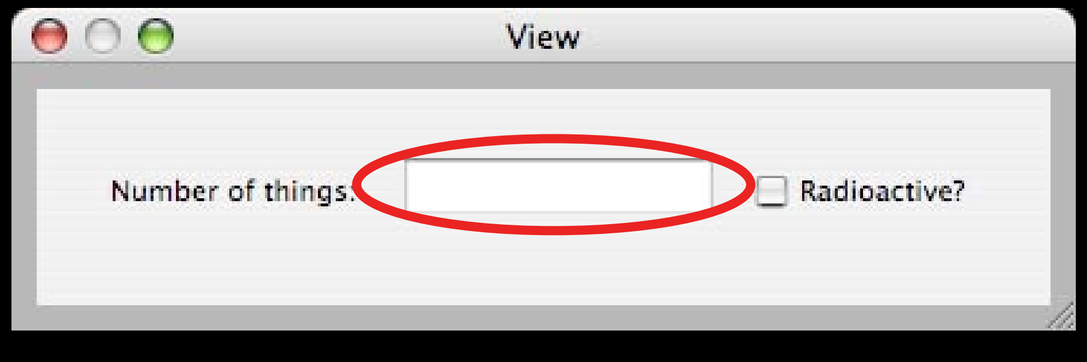
Drag objects from IB palettes onto your View

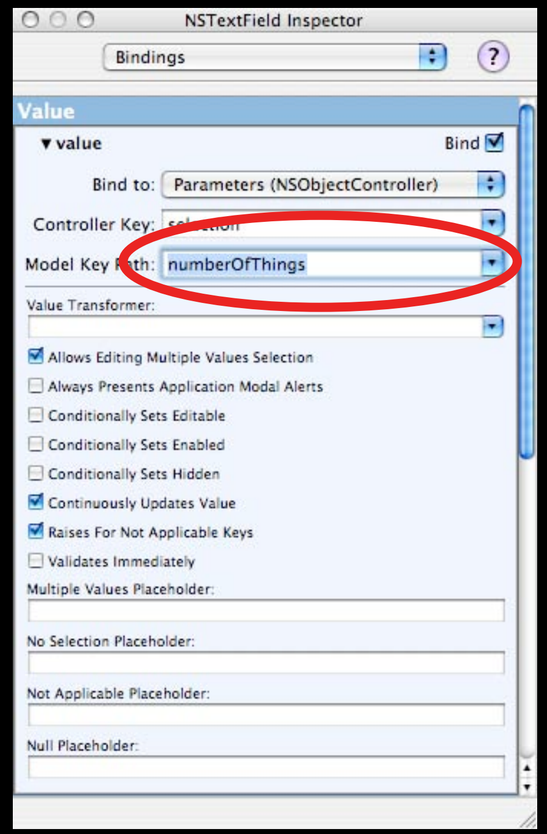


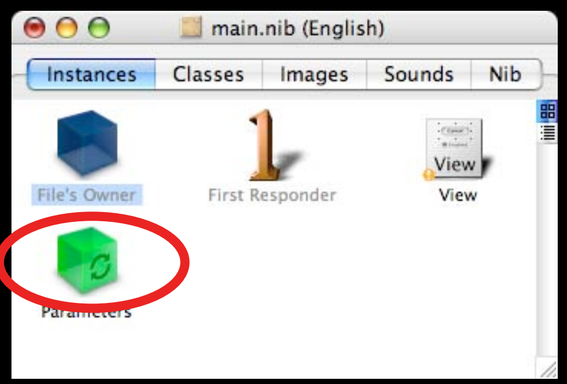


In Interface Builder

Set value bindings

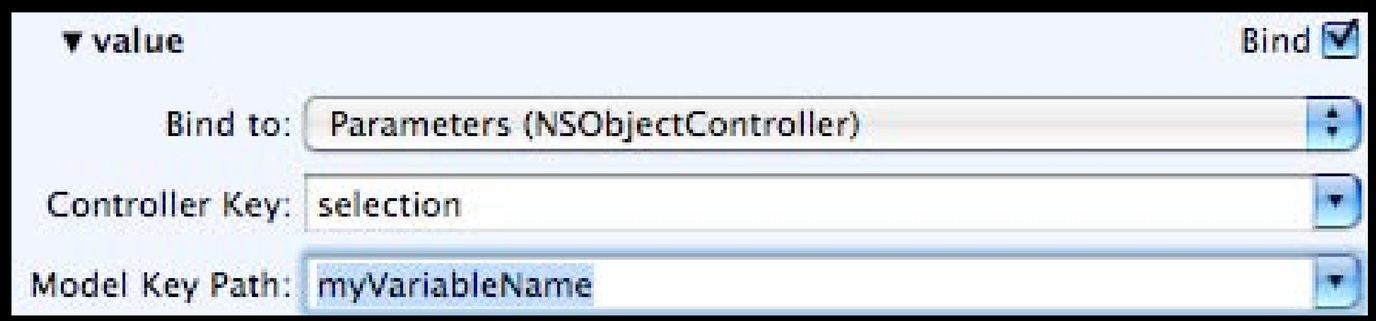






General Principles

- Bind to Parameters (NSObjectController)
- Controller key: selection
- Model Key Path: A variable name of your choice
  - 3 things MUST match
    - Model Key Path (IB)
    - Parameter key (Info.plist)
    - Variable name (Shell script)

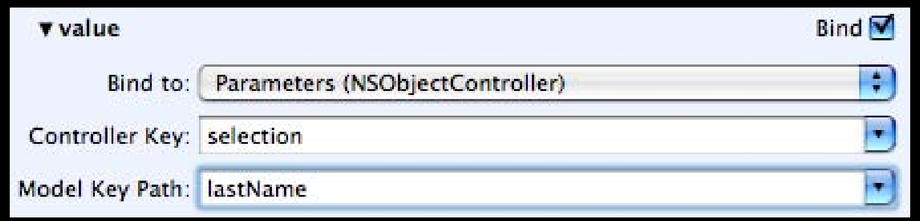


Binding Simple IB Objects

- Text fields: value
- Checkboxes:
- Sliders
- Radio Buttons
- Popups:

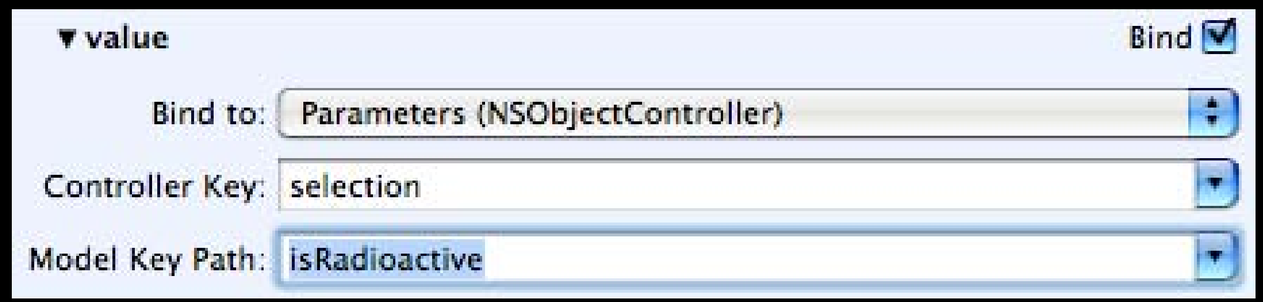
Text fields

- Bind value
- Choose Continuously Updates Value
  - Otherwise the value is only updated if you hit “return”



Checkboxes

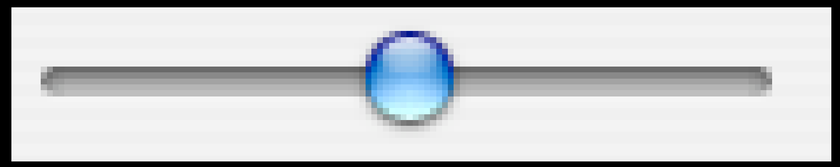
- Bind value
- Your script sees a 0 (unchecked) or 1 (checked)



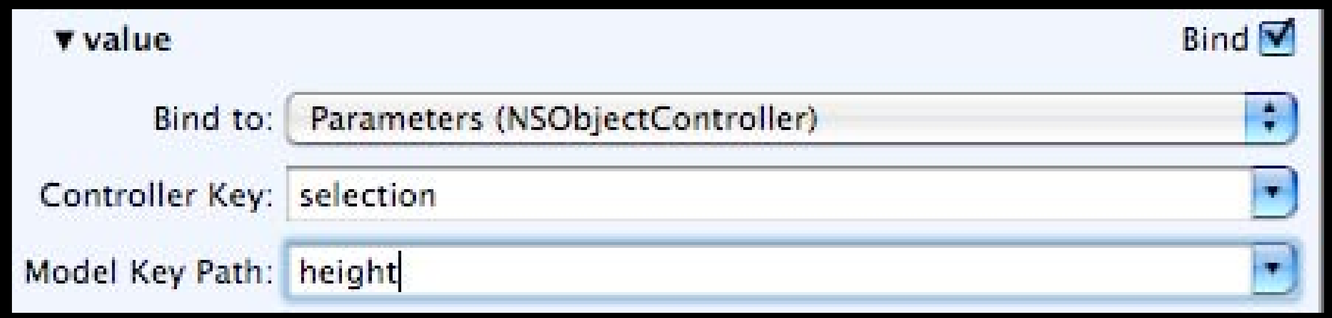
bash:

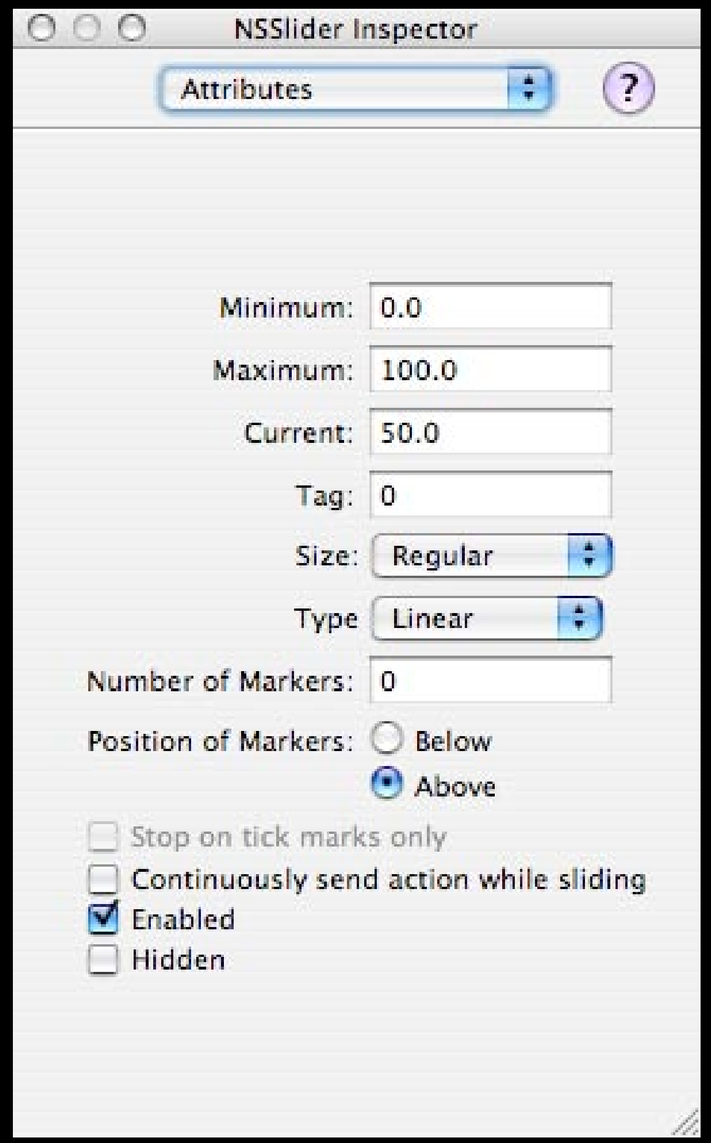
```
if (( $isRadioactive )); then rm /dangerous/file; fi
```

Sliders



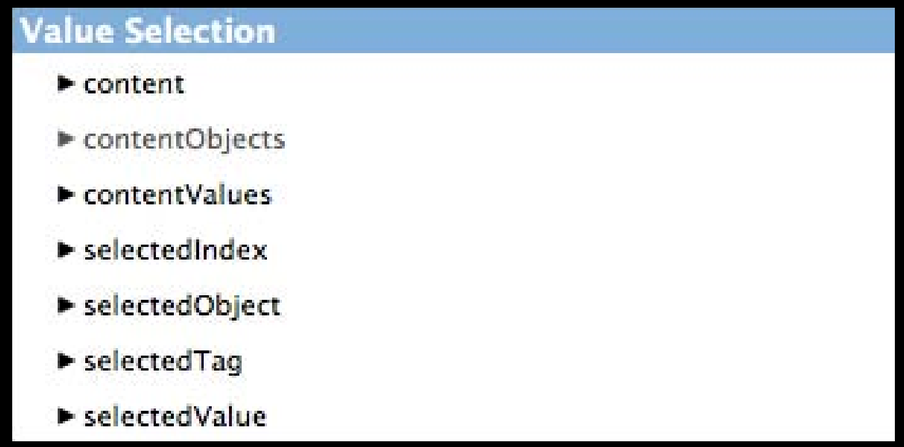
- Bind to value
- You get a number according to the slider's position





Radio buttons

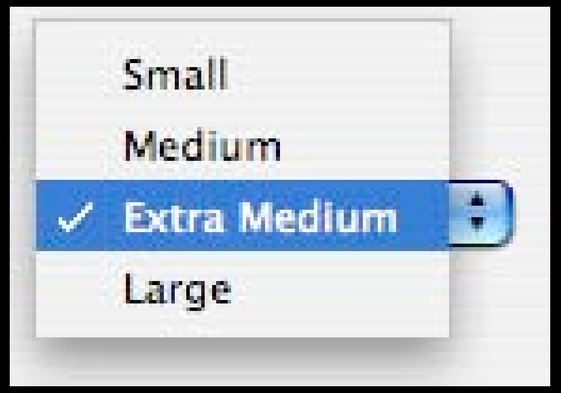
- Choices! What to bind to?
- selectedIndex
  - 0, 1, 2
- selectedValue
  - Canada, USA, Mexico



IB Tip: Option-Drag to create more buttons

Popups

- More choices!
- selectedIndex
  - 0,1,2,3
- selectedValue
  - Small, Medium, Extra Medium, Large



/usr/bin/automator

Run workflows from the command line

http://developer.apple.com/wwdc2005/

/usr/bin/automator

Run a workflow from the command line

```
automator /Library/Workflows/MyWorkflow.workflow
```

- No user interaction allowed

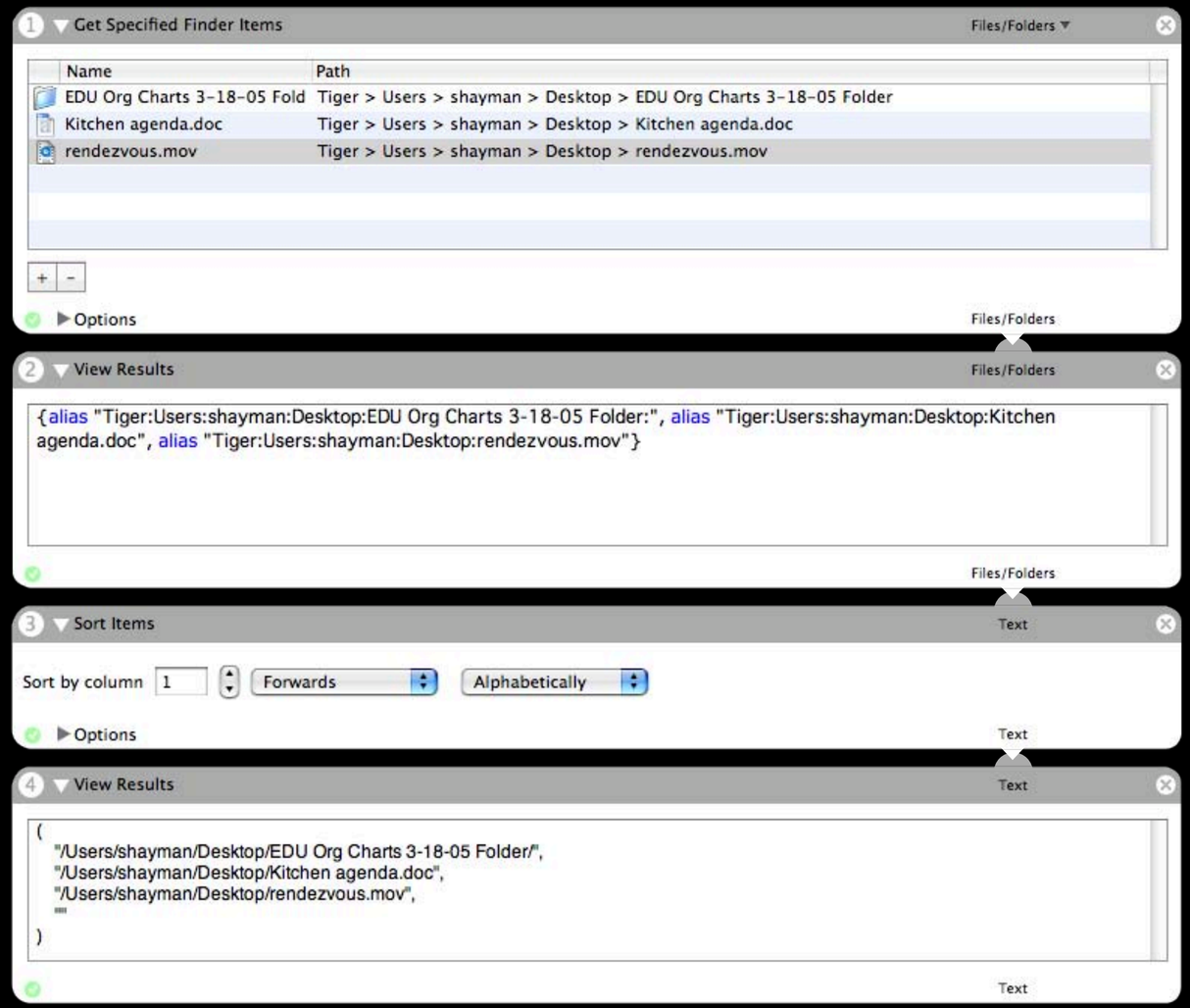
Tips

http://developer.apple.com/wwdc2005/

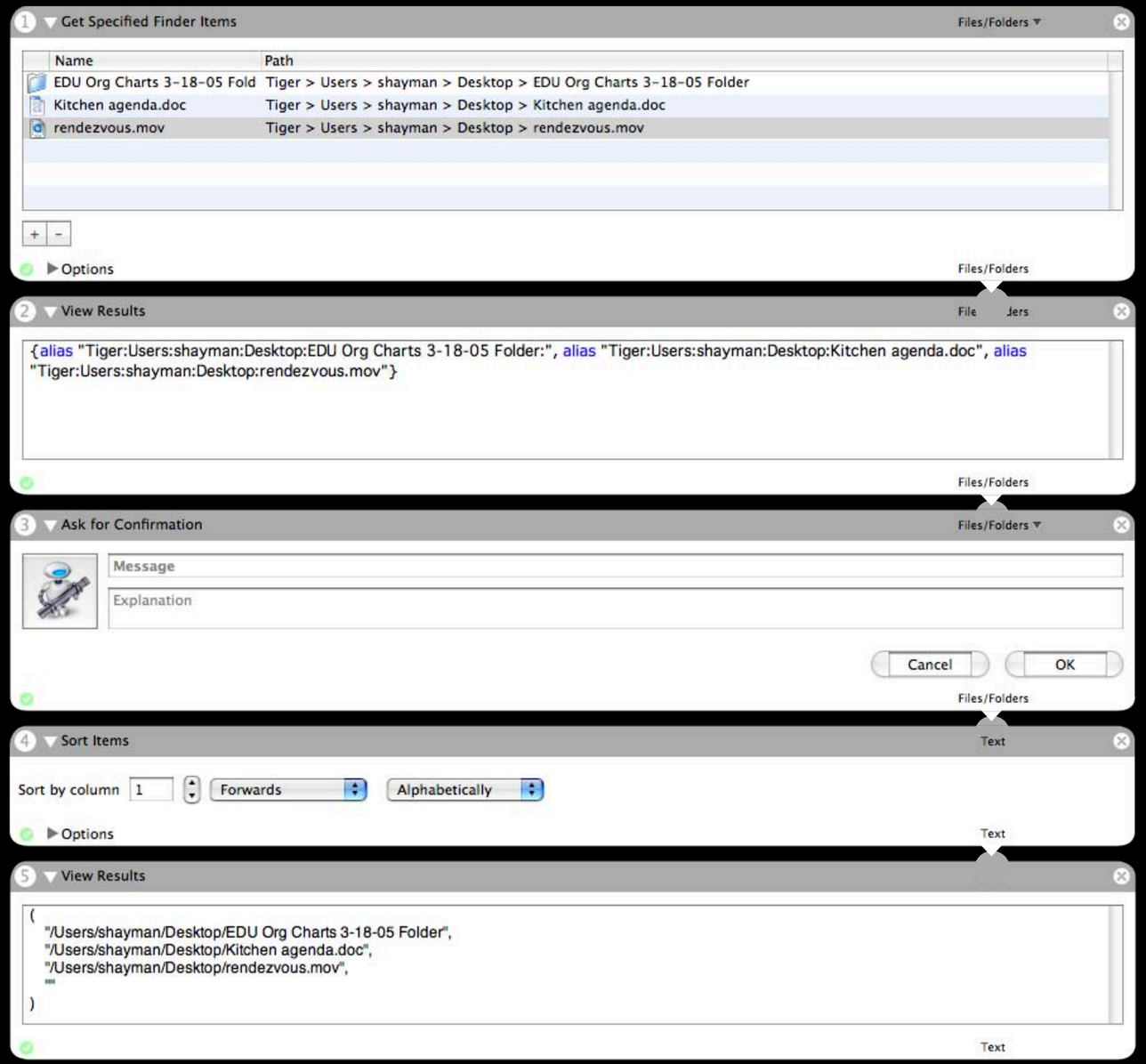
Debugging Your Workflow

- There isn’t a debugger
- But…

Use “View Results” Liberally!



Use “Ask for Confirmation” too!



A Handy Applescript

- Automator itself is Apple Scriptable
- How about a script that inserts “View Results” and “Ask for Confirmation” after every action in your script

Debugging Scripts

- Standard Error is normally discarded
- Try redirecting it somewhere
  - sh:

```
exec 2>/dev/console
```

  - perl:

```
open(STDERR, “>/dev/console”);
```

- Now you can write debug messages and watch for them in Console.app

```
echo About to sort files 1>&2
print STDERR “Opening portal\n”;
```

Debugging sh scripts

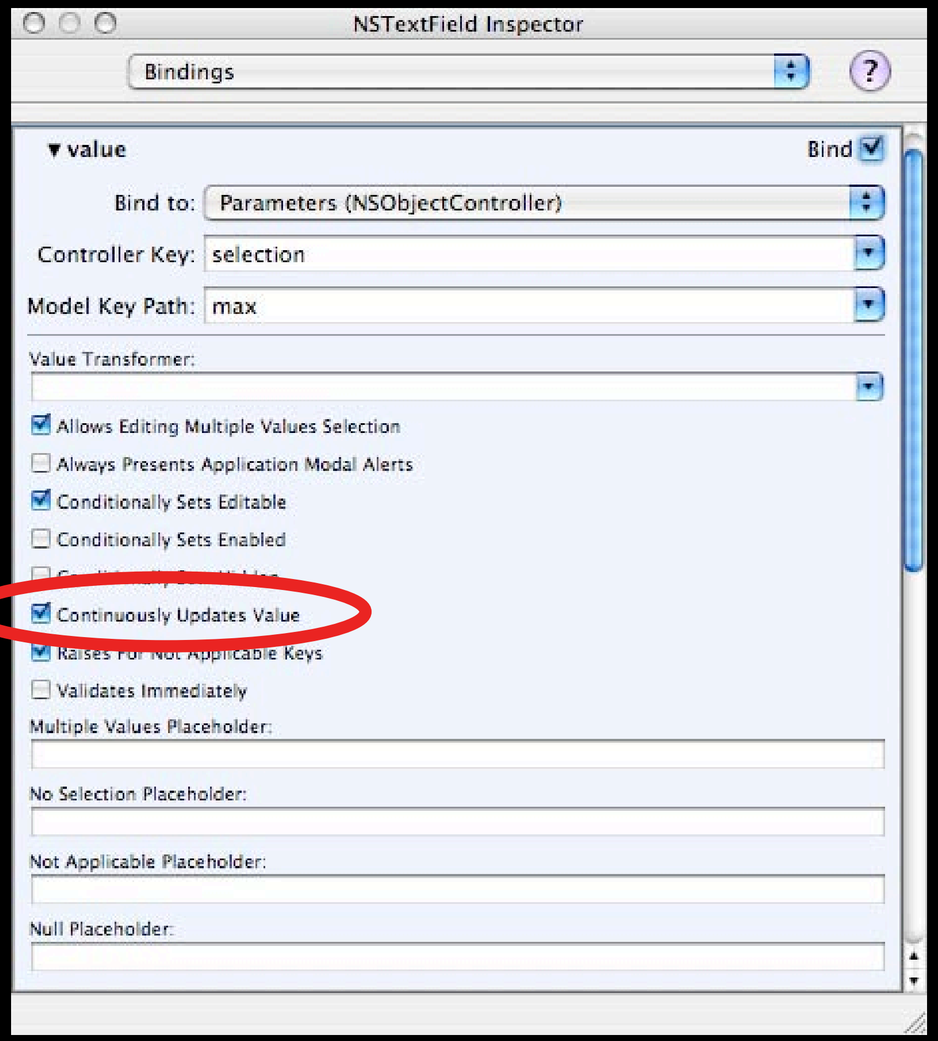
- Try this

```
# Log statements as they are executed to the console

set -xv
exec 2>/dev/console

# Print all our environment variables, so we can check bindings
printenv
```

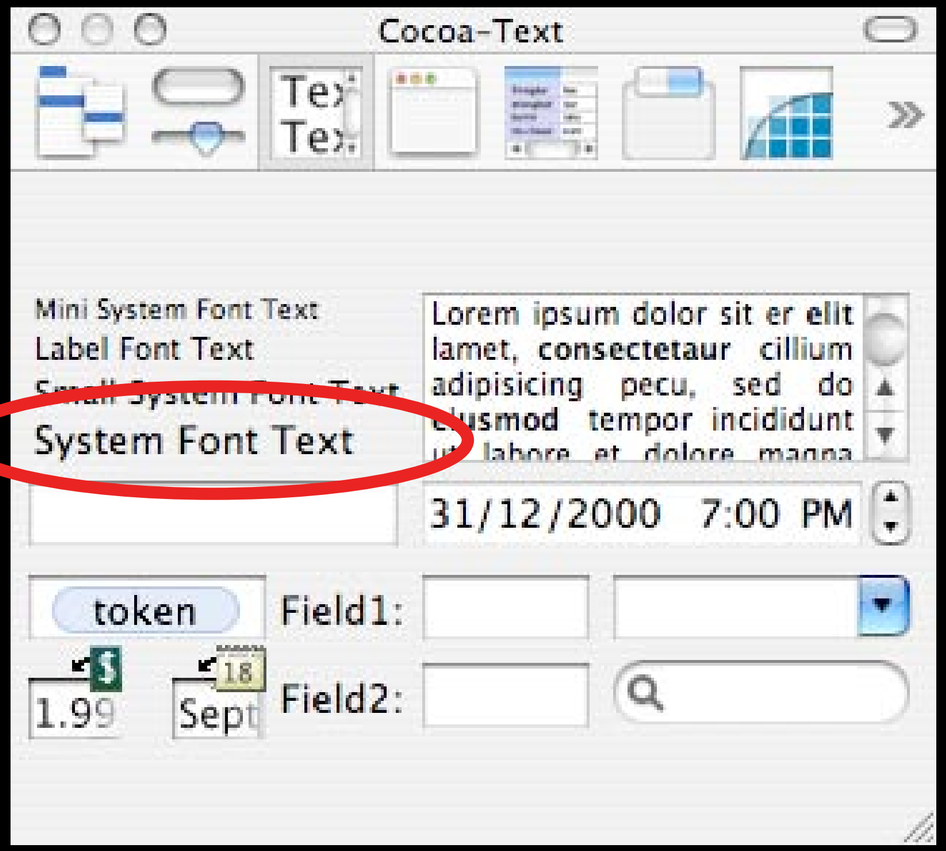
Possible Demo of Script that Inserts “View Results”

http://developer.apple.com/wwdc2005/

Tips in IB



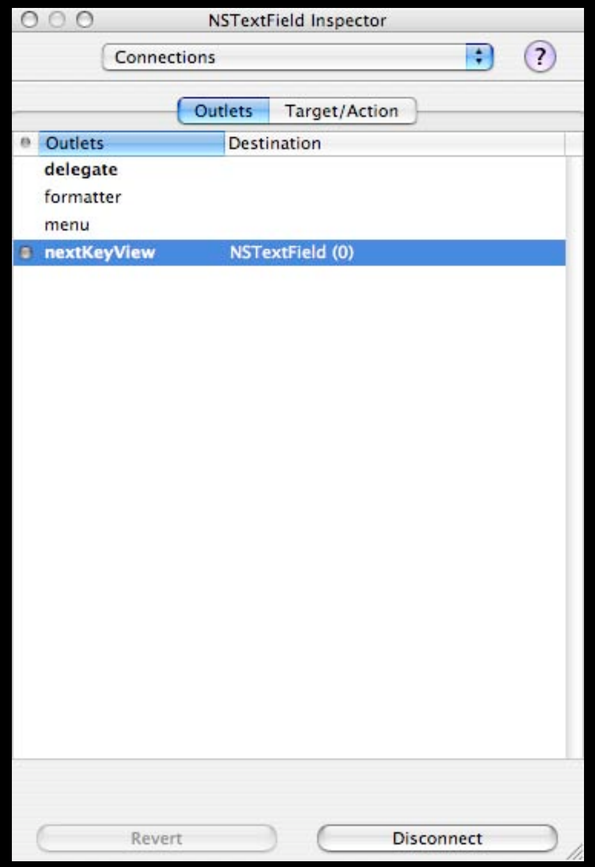
- For text fields,
- Check “Continuously Updates Value”
- Value will get set whether or not the user hits return



Use Small System Font

- Otherwise amlint will complain when you build your action

Connect text fields



- Connect the **nextKeyView** outlet of each text field
- Otherwise amlint will complain

Set the Bundle Identifier

- Info.plist
- Something unique for each of your actions
- <key>CFBundleIdentifier</key>
  - <string>com.apple.shayman.sortitems</string>

Spaces in file names

- Watch out! Quote things carefully!

```
for f in “$@”; do
    ls -l “$f”
done
```

For More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference